Proceedings of the 4th International Conference on Functional Materials and Civil Engineering
DOI: 10.54254/2755-2721/2026.38198

Research on Lightweight Industrial Large Models for Fault Diagnosis on Edge Devices

Yizhou Cao

Department of Mechanical and Aerospace Engineering, University of Manchester, Manchester, UK
caoyizhou1127@163.com

Abstract. Industrial large models can extract features from industrial data, transfer learned knowledge and adapt to different fault diagnosis tasks. However, their massive parameter counts, high memory usage, and expensive computational costs make direct deployment on resource-constrained edge devices challenging. This paper reviews the current applications and deployment challenges of industrial large models in fault diagnosis and provides an in-depth discussion of six major lightweight techniques: model pruning, quantization, knowledge distillation, lightweight architecture design, parameter-efficient fine-tuning, and hybrid optimization. The paper analyzes the advantages, limitations, and applicability to different hardware of these techniques from dimensions such as diagnostic accuracy, model size, inference latency, memory usage, and energy consumption. In addition, based on the computing capabilities of various types of industrial equipment, this paper proposes a resource-aware deployment framework: for low-power terminals, models with high compression ratios are recommended; for medium-power devices, models that are moderately optimized and combined with an edge-to-cloud collaboration mechanism are suitable; while for complex diagnostic tasks, larger-scale models can be deployed on high-performance edge servers or cloud platforms. Overall, this combination may improve the accuracy and efficiency of edge-based industrial fault diagnosis.

Keywords: industrial large models, fault diagnosis, edge computing, model lightweighting, model compression

1. Introduction

In recent years, with the development of smart manufacturing, industrial equipment is becoming increasingly complex and producing a lot of data, including vibration signals, images, temperature records, and maintenance logs. Accurate fault diagnosis is important because equipment failure may cause production interruptions, increase maintenance cost, and even create safety risks [1].

Traditional fault diagnosis methods usually rely on expert knowledge, artificially designed features or deep learning models for specific tasks. Although these methods can achieve better results on specific devices, their diagnostic performance may decrease when device type, operating condition, or fault mode changes. Industrial large models can help improve these limitations. However, its large parameters, high computing volume and high memory occupation make it difficult to meet the real -time and power consumption requirements of edge devices [2, 3].

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38198

Therefore, this paper reviews six lightweighting techniques and proposes resource-aware deployment strategies for edge devices with different computational capabilities. This article will also introduce the complete deployment process and compare the deployment strategies of low computing power, medium computing power and high computing power edge devices. This study aims to balance diagnostic performance with development efficiency on edge devices.

2. Industrial large models and fault diagnosis

2.1. Potential of industrial large models

Industrial large models are Transformer-based models designed for industrial applications. After pretraining on large datasets, these models can adapt to different industrial tasks [4]. Compared to traditional deep learning models, large models possess stronger capabilities in multitasking, generalization, communication, and reasoning. They can not only identify fault categories but also describe fault characteristics, analyze potential causes, and generate maintenance recommendations [5].

The development of industrial artificial intelligence is generally shifting from relying on manually designed features and task-oriented neural networks to pre-trained models and large-scale Transformer model. Based on data modality, industrial foundation models can be divided into language, vision, time-series, and multimodal models, which respectively process documents, images, sensor signals, or combinations of heterogeneous data [2].

2.2. Industrial fault diagnosis tasks

Industrial fault diagnosis is not just about determining whether equipment is abnormal, but also includes stages such as fault detection, fault isolation, fault identification and fault classification (Figure 1). Its goal is to determine the location, type and severity of the fault, and then provide a basis for subsequent maintenance decisions [6]. Common diagnostic data includes vibration, acoustic emission, temperature, pressure, torque, current and thermal imaging images, among others. Among these, vibration signals have the advantages of relatively lower acquisition costs and higher reliability, which makes them one of the most commonly used types of data in rotating machinery fault diagnosis [7].

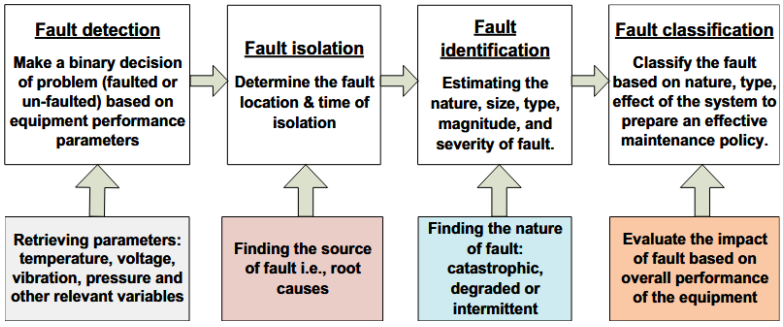


Figure 1. Main stages of industrial fault diagnosis [8]

Large language models can also combine vehicle operation data, maintenance records, and technical documents to assist with predictive maintenance and fault diagnosis [9]. However, such applications rely on diverse and high-quality data and require relatively high computational and deployment costs, which limits the further application of models on resource-constrained devices.

DOI: 10.54254/2755-2721/2026.38198

3. Challenges of edge deployment

Industrial fault diagnosis requires timely processing of sensor data to act, and compared with cloud-based diagnosis, direct edge deployment processes data close to industrial equipment, which reduces communication latency, allowing diagnostic services to be maintained more stably. However, large industrial models require substantial computing resources and energy consumption, which can also lead to a mismatch between the requirements for running the models and the conditions of actual edge devices.

3.1. Characteristics of edge devices

Edge devices are located near sensors or industrial machines and can process vibration, current, temperature and image data locally, thereby reducing communication latency, bandwidth consumption, and reliance on remote cloud servers [10]. The heterogeneity of hardware and software is also a major characteristic of edge devices. Industrial sites may simultaneously have devices using different processors, operating systems, memory capacities and precision. Therefore, the deployment of edge devices requires models to have the ability to adapt to heterogeneous hardware, while maintaining stability and real-time performance of fault diagnosis.

3.2. Deployment challenges

The development of industrial large models in edge devices involves several technical challenges. The resource requirements of industrial large models may exceed the computing, memory, and energy capabilities of edge devices, thereby leading to higher inference latency and memory consumption [11]. Secondly, edge deployment needs to meet the real-time requirements of industrial fault diagnosis. Therefore, inference latency must also be considered during deployment, and the worst-case response time must be evaluated. When local resources are insufficient, computing tasks can be offloaded to edge servers or cloud platforms, but task offloading introduces additional communication latency, and makes diagnostic performance dependent on network conditions and external computing resource [10]. Local processing can reduce the transmission of sensitive production data, although distributed edge devices may introduce additional security risks [11]. Changes in operating conditions and component wear may also cause shifts in the data distribution and reduce diagnostic reliability. These challenges make direct deployment difficult and create a need for hardware-aware lightweight techniques.

3.3. Balancing diagnostic and deployment objectives

The design of edge-side fault diagnosis systems needs to strike a balance between diagnostic performance and deployment efficiency. Although increasing model complexity can enhance feature representation capabilities and improve fault classification accuracy, it also brings higher computational overhead, memory usage, and inference latency. Therefore, an effective deployment strategy should reduce computational complexity and ensure real-time fault detection while maintaining sufficient diagnostic accuracy [12].

4. Lightweight technologies for industrial large models

Existing lightweighting techniques can generally be divided into model pruning, model quantization, knowledge distillation, parameter-efficient fine-tuning, lightweight architecture design and hybrid

DOI: 10.54254/2755-2721/2026.38198

optimization. These techniques are applied at different stages of model development and deployment, and thus each has its own advantages and limitations.

4.1. Model pruning

To deploy models with a massive number of parameters on resource-constrained edge devices, model compression is necessary. Model pruning is a core technique for model lightweighting, commonly applied to large-scale models and deep learning networks. It operates by removing parameters, neurons, channels, or network layers that have minimal impact on the model's output, thereby reducing model complexity while striving to maintain accuracy.

According to pruning granularity, pruning methods can be divided into unstructured pruning and structured pruning [13]. Unstructured pruning simply directly removes individual weight parameters, and can usually achieve a high compression ratio, but it produces sparse matrices. Structured pruning removes an entire channel, convolution kernel, filter or layer at a time, and the resulting dense model makes the compressed model more compatible with general-purpose edge device hardware. However, the accuracy loss at the same compression ratio is greater than that of unstructured pruning.

Zhu and Gupta proposed an automated progressive pruning method that gradually increases the model's sparsity rate during training, allowing the model to recover some of the performance lost due to parameter removal [14]. However, the pruning ratio should not be increased without considering model sensitivity. Zhang et al. proposed PAOLTransformer, which applies adaptive structured pruning to an aero-engine RUL prediction model. In the PAOLTransformer experiments, predictive performance improved when moderate pruning removed redundant structures, whereas pruning rates exceeding approximately 40% resulted in a significant performance decline. This indicates that excessive pruning may remove components necessary for characterizing equipment degradation patterns [15].

4.2. Quantization

Model quantization involves converting model weights and activations from high-precision floating-point formats to low-precision numerical formats for storage and computation. This reduces GPU memory usage, accelerates inference, and lowers power consumption.

Model quantization can generally be implemented via Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ converts a pre-trained model—with fixed parameters—into a low-precision format by performing forward inference on a calibration dataset. In contrast, QAT simulates quantization during the training process and exports a low-bit model upon completion. QAT usually retains more accuracy because the model learns to account for quantization errors during training [16].

Low-bit quantization can reduce more memory usage and computational cost, but it may also introduce larger numerical errors. Mixed-precision quantization uses higher precision for sensitive layers and lower precision for other layers [17, 18].

4.3. Knowledge distillation

Knowledge distillation is a model compression technique designed to transfer knowledge from a large and complex teacher model to a smaller student model. Unlike traditional training methods that learn solely from ground-truth labels, knowledge distillation allows the student model to mimic

DOI: 10.54254/2755-2721/2026.38198

the outputs or internal representations of the teacher model. Because these outputs contain additional information about the relationships between different classes, the student model can learn a richer decision structure than by using hard labels alone. Therefore, knowledge distillation can reduce model size and computational requirements while retaining a substantial portion of the teacher model's predictive performance [16, 19].

Depending on the information transmitted, distillation can be response-based, feature-based, or relation-based, utilizing teacher outputs, intermediate representations, or the relationships between samples and features, respectively, as shown in Figure 2 [20].

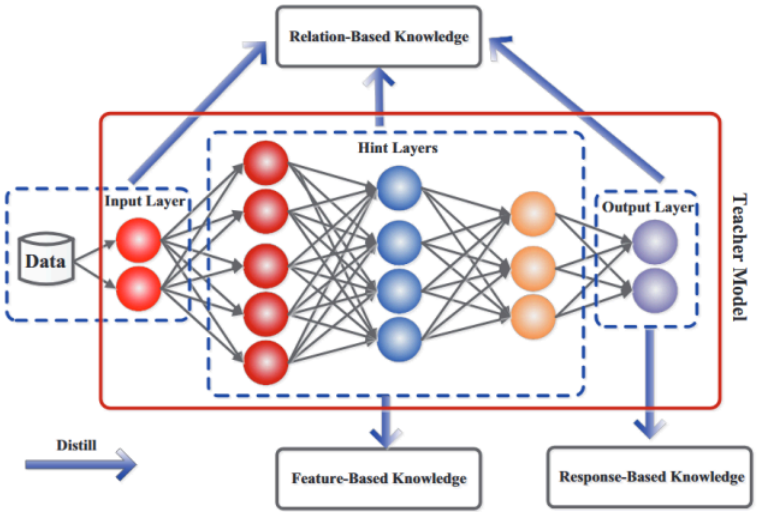


Figure 2. Main types of knowledge transferred in knowledge distillation [20]

In industrial applications, knowledge distillation can transfer the diagnostic capabilities of large teacher models to small student models suitable for edge deployment. For example, Liu et al. developed KD-LightNet for industrial defect detection and used knowledge distillation to improve the performance of the lightweight student network, which shows that the application potential of this technology in resource-constrained industrial inspection systems [21].

However, knowledge distillation still faces many challenges. Distillation performance depends on the quality of the teacher model, the capabilities of the student model, and the selected objectives. Significant architectural differences may hinder knowledge transfer, while feature- and relation-based methods may increase training complexity. Therefore, in actual edge deployment, it is also necessary to consider training costs, diagnostic accuracy, model size and the computing capabilities of edge devices and find a balance among them [20].

4.4. Parameter-efficient fine-tuning

Parameter-efficient fine-tuning freezes most of the large models during training, training only a small number of newly added parameters without updating the entire model. Compared with full fine-tuning, it uses less GPU memory and trains faster [22]. Existing PEFT methods can be broadly divided into four categories: additive, selective, reparameterization-based and hybrid. Representative techniques include Adapter Tuning, Prompt Tuning, Prefix Tuning and low-rank adaption (LoRA) [23].

In industrial applications, PEFT has the capability to handle multi-task scenarios, allowing a shared model to be applied to different diagnostic tasks. Secondly, PEFT can significantly reduce the

DOI: 10.54254/2755-2721/2026.38198

GPU memory and computational resource requirements for fine-tuning. However, a reduction in trainable parameters does not necessarily mean a proportional reduction in computation or memory usage, because the frozen backbone network and its intermediate activations still need to be processed [22]. Therefore, PEFT mostly reduces adaptation costs and storage costs, rather than directly compressing the backbone model. It should be combined with pruning, quantization, or other lightweight techniques on resource-constrained edge devices

4.5. Lightweight architecture design

Lightweight architecture reduces the number of parameters and computations at the structural level when designing the network, rather than compressing it after training. Common techniques include depthwise separable convolution and efficient attention mechanisms. These structures can reduce parameters and computation while maintaining feature extraction capabilities. For Transformer-based models, using local, sparse, or linear attention mechanisms can reduce the computational and memory overhead caused by standard self-attention mechanisms. Among them, hardware-aware NAS can include actual latency, energy consumption, and hardware constraints into the search objectives, thereby making the generated architectures more suitable to the required edge device environment [16, 24].

This approach is widely used for edge-based fault diagnosis involving large-scale industrial models. Ma et al. proposed a multi-objective architecture search method that jointly considers accuracy and actual inference latency on the target device. In an experiment involving fault diagnosis for rotating machinery, the generated model achieved 100% diagnostic accuracy with a size of only 82 KB and a computational cost of 6.08 M FLOPs; it also attained 99.09% accuracy on another dataset [25]. However, architecture exploration requires substantial training resources, and lightweighting itself has an accuracy ceiling, so lightweight architectures still need to strike a balance between factors such as accuracy and hardware compatibility.

4.6. Hybrid optimization

A single lightweighting technique often struggles to simultaneously meet various operational requirements such as model size, memory usage, inference latency, efficiency and diagnostic accuracy. Therefore, hybrid strategies allow multiple techniques to complement each other and strike a balance among different objectives. For example, combining knowledge distillation with quantization, distillation compensates for the accuracy loss caused by quantization, while quantization reduces inference overhead. Pruning first removes redundant weights, and quantization can further reduce the data precision of the remaining parameters. PEFT can also be combined with quantization techniques, requiring fewer training parameters and less memory, and making it better suited to specific diagnostic tasks [16, 22, 24].

Hybrid optimization allows multiple techniques to be stacked to achieve a greater degree of model slimming, meeting the memory and computing power constraints of edge devices. However, excessive stacking also causes accuracy to decline. At the same time, hybrid optimization also makes training take longer. Therefore, when stacking compression, accuracy should also be evaluated, and an appropriate scheme should be comprehensively considered based on the evaluation metrics and actual conditions of the target edge platform.

DOI: 10.54254/2755-2721/2026.38198

5. Edge deployment strategies and process

Edge deployment is not simply copying a trained model to a device, it requires first clarifying the specific task and hardware requirements, then completing model compression, format conversion, hardware testing and continuous monitoring. Since different computing devices can produce different training results, different strategies should be used accordingly.

5.1. Deployment pipeline

The first step of the deployment process is to analyze the diagnostic task, input data, latency requirements and hardware resources. Then, select an appropriate pretrained model based on the amount of data for the specific task and the resources available for training, and perform full fine-tuning or parameter-efficient fine-tuning to adapt it to the target device. After adaptation, compression can be performed according to the computing power or memory requirements of the target device, using pruning, quantization, knowledge distillation or hybrid optimization methods. Then, the optimized model is converted into a hardware-compatible inference format and integrated with the data acquisition and preprocessing modules. Before deployment, the diagnostic performance on the actual device should be evaluated, because reductions in the number of parameters or floating-point operations do not necessarily improve latency or energy consumption [11, 24]. When the model evaluation results meet the expected accuracy, memory usage, latency and reliability metrics, it can be deployed for online monitoring. Jia et al. demonstrated this process by adapting a pretrained diagnostic model to resource-constrained terminals through lightweight task-specific customization [26].

5.2. Strategy for low-compute devices

Based on available memory, processing capability, hardware acceleration, and power budget, industrial edge platforms can be classified into low, medium, and high computing power tiers. For different tiers, corresponding lightweighting and deployment strategies should be selected [11].

Low-compute devices are typically subject to strict limitations in terms of memory, processing capability, storage space and energy consumption; therefore, it is not possible to deploy complete industrial large models on such devices. A more suitable approach is to use lightweight CNNs or distilled student models, combined with structured pruning and INT8 quantization [16, 24]. These devices can perform continuous data processing, anomaly detection and fault classification. When encountering uncertain samples, the filtered data features can be transmitted to edge servers or cloud platforms. This model enables low-compute devices to maintain real-time local monitoring, while also leveraging external computing resources to handle more demanding diagnostic tasks [27].

5.3. Strategy for medium-compute devices

Compared with low-compute devices, medium-compute devices, such as industrial gateways and edge device platforms equipped with GPUs or NPUs, have larger memory capacities and stronger processing capabilities. So, they can support compressed diagnostic models while maintaining accuracy. For such hardware, structured pruning can be combined with INT8 or FP16 quantization techniques to reduce memory usage and computational overhead while maintaining a hardware-friendly model architecture [24].

DOI: 10.54254/2755-2721/2026.38198

These devices are suitable for adopting a local-first edge-cloud collaborative strategy. Routine fault detection and assessment are performed locally, while complex model training and updates are performed on cloud servers [27]. The deployment system can also assign tasks to different models for processing based on task complexity, simple ones to lightweight models, and those with high uncertainty or complexity to larger-scale models or cloud processing. This strategy can achieve a balance among diagnostic accuracy, resource consumption and communication dependency.

5.4. Strategy for high-compute devices

High-compute devices can support large diagnostic models using lighter compression strategies, integrate multiple types of heterogeneous information from vibration, acoustics, thermal data, etc., and perform complex tasks such as simulation training, multimodal data fusion, etc. Garcia et al. pointed out that IoT and big data infrastructure can support large-scale data collection and real-time analysis, while the edge-to-cloud architecture can handle massive industrial data streams [28].

Such hardware platforms can retain complete models or moderately compressed versions to ensure diagnostic capabilities. However, in actual deployment there is no need to migrate all tasks to the cloud. Somu and Dasappa proposed that edge deployment or cloud deployment should be selected according to the computing capabilities of edge devices [27]. This can enable rapid local response while also leveraging the powerful computing and storage capabilities of high-compute infrastructure, but factors such as data privacy and network reliability still need to be considered.

6. Conclusion

Industrial large models can learn from heterogeneous data and transfer knowledge and adapt to and be deployed on different types of equipment. However, their enormous parameter scale, high memory requirements, and high computational costs make direct deployment on edge devices challenging. This paper reviews six major lightweight techniques, including model pruning, quantization, knowledge distillation, lightweight architecture design, parameter-efficient fine-tuning, and hybrid optimization. These techniques reduce deployment costs from different dimensions, but their effectiveness depends on the target model, diagnostic task, hardware platform, and the required balance between accuracy and efficiency.

This paper further explores a resource-aware deployment framework tailored to industrial devices with varying computing capabilities. Low-power devices require highly compressed models and simple local inference; medium-power devices can adopt an approach that combines moderate compression with edge-cloud collaboration; while high-power platforms can support larger-scale models while performing complex tasks such as multimodal analysis, model updates, and remaining useful life prediction. Future research should focus on hardware-aware optimization, adaptive model selection, multimodal diagnostic models, and evaluation in real industrial environments. Overall, combining lightweight techniques with hardware-specific deployment strategies is crucial for achieving accurate, efficient, and reliable industrial fault diagnosis at the edge.

References

[1] Zhou, A. Y., & Barati Farimani, A. (2024). FaultFormer: Pretraining transformers for adaptable bearing fault classification. *IEEE Access, 12*, 70719–70728.

[2] Chen, J., et al. (2024). Towards general industrial intelligence: A survey of continual large models in industrial IoT. Dec. 27. arXiv: arXiv:2409.01207.

[3] Li, Y., et al. (2024). Large language models for manufacturing. Oct. 28. arXiv: arXiv:2410.21418.

DOI: 10.54254/2755-2721/2026.38198

[4] Zhou, J., et al. (2024). Industrial large model: A survey. *MATEC Web Conf., 401*, 10009.
[5] Wang, H., Li, C., Li, Y. F., & Tsung, F. (2024). An intelligent industrial visual monitoring and maintenance framework empowered by large-scale visual and language models. *IEEE Transactions on Industrial Cyber-Physical Systems, 2*, 166–175.
[6] Sahu, A. R., Palei, S. K., & Mishra, A. (2024). Data-driven fault diagnosis approaches for industrial equipment: A review. *Expert Systems, 41*(2), e13360.
[7] Yan, W., Wang, J., Lu, S., Zhou, M., & Peng, X. (2023). A review of real-time fault diagnosis methods for industrial smart manufacturing. *Processes, 11*(2), 369.
[8] Sahu, A. R., & Palei, S. K. (2020). Real-time fault diagnosis of HEMM using Bayesian Network: A case study on drag system of dragline. *Engineering Failure Analysis, 118*, 104917.
[9] Raza, M., Jahangir, Z., Riaz, M. B., Saeed, M. J., & Sattar, M. A. (2025). Industrial applications of large language models. *Sci Rep, 15*(1), 13755.
[10] Cruz, P., Achir, N., & Viana, A. C. (2022). On the edge of the deployment: A survey on multi-access edge computing. *ACM Comput. Surv., 55*(5), 99:1-99:34.
[11] Wang, T., Guo, J., Zhang, B., Yang, G., & Li, D. (2025). Deploying AI on edge: Advancement and challenges in edge intelligence. *Mathematics, 13*(11), 1878.
[12] Leite, D., Andrade, E., Rativa, D., & Maciel, A. M. A. (2025). Fault detection and diagnosis in Industry 4.0: A review on challenges and opportunities. *Sensors, 25*(1), 60.
[13] Vadera, S., & Ameen, S. (2022). Methods for pruning deep neural networks. *IEEE Access, 10*, 63280–63300.
[14] Zhu, M., & Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. Nov. 13. arXiv:1710.01878.
[15] Zhang, X., Sun, J., Wang, J., Jin, Y., Wang, L., & Liu, Z. (2023). PAOLTransformer: Pruning-adaptive optimal lightweight Transformer model for aero-engine remaining useful life prediction. *Reliability Engineering & System Safety, 240*, 109605.
[16] Musa, A., Kakudi, H. A., Hassan, M., Hamada, M., Umar, U., & Salisu, M. L. (2025). Lightweight deep learning models for edge devices—A survey. *International Journal of Computer Information Systems and Industrial Management Applications, 17*, 18–18.
[17] Kuzmin, A., Nagel, M., van Baalen, M., Behboodi, A., & Blankevoort, T. (2023). Pruning vs quantization: Which is better? *Advances in Neural Information Processing Systems*, Curran Associates, Inc., pp. 62414–62427.
[18] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., & Keutzer, K. (2019). HAWQ: Hessian AWare quantization of neural networks with mixed-precision. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302.
[19] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531.
[20] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *Int J Comput Vis, 129*(6), 1789–1819.
[21] Liu, J., Li, H., Zuo, F., Zhao, Z., & Lu, S. (2023). KD-LightNet: A lightweight network based on knowledge distillation for industrial defect detection. *IEEE Transactions on Instrumentation and Measurement, 72*, 1–13.
[22] Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. arXiv:2403.14608.
[23] Xu, L., Xie, H., Qin, S. J., Tao, X., & Wang, F. L. (2026). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 48*(6), 6107–6126.
[24] Lai, N., Dewi, D. A., Maidin, S. S., Xiao, W., Zhao, S., & Hu, Q. (2026). A comprehensive review of lightweight deep learning models for edge computing with future directions. *Discov Computing, 29*(1), 110.
[25] Ma, S., Sun, H., Gao, S., & Zhou, G. (2023). A real-time mechanical fault diagnosis approach based on lightweight architecture search considering industrial edge deployments. *Engineering Applications of Artificial Intelligence, 123*, 106433.
[26] Jia, X., Qin, N., Huang, D., Du, J., & Wang, T. (2025). Industrial large-scale diagnostic model with lightweight customized deployment for distributed multiple non-IID diagnostic tasks. *IEEE Sensors Journal, 25*(14), 27043–27055.
[27] Somu, N., & Dasappa, N. S. (2025). An edge-cloud IIoT framework for predictive maintenance in manufacturing systems. *Advanced Engineering Informatics, 65*, 103388.
[28] Garcia, J., Rios-Colque, L., Peña, A., & Rojas, L. (2025). Condition monitoring and predictive maintenance in industrial equipment: An NLP-assisted review of signal processing, hybrid models, and implementation challenges. *Applied Sciences, 15*(10), 5465.